**Logo, company name

Description automatically generated**

**JK Lakshmipat University, Jaipur**

**Institute of Engineering and Technology (IET)**

**CS1117 Computer Organization and System**

**Lab File**

SUBMITTED BY

**Akshat singh**

22018

SUBMITTED TO

**Mr. Divanshu Jain**

**EXPERIMENT – 3**

**OBJECTIVE**

WAP to ascending sort 5 Numbers.

**THEORY**

**The code used is in RED**

**The registers are in BLUE**

**The statements are in GREEN**

**---------------------------------------- CODE SEGMENT-------------------------------**

1. **CODE SEGMENT : - We are using the ASSUME argument to tell the assembler which segment register we are going to use to access a segment**

**; - By writing (ASSUME CS:CODE we are specifying (CS) as CODE**

1. **START :**

**; - This is a simple argument to tell the assembler that the code starts from here.**

**-------------------------------------CODE TO SORT-------------------------------------**

1. **We start by writing the (MOV) instruction to move the value (2000H) into (AX).**

**Which is the primary accumulator, and it is used for arithmetic instructions. = (MOV AX,2000H)**

1. **Next, we use (MOV) again to move the value 2000H which is now in AX to (DS) = (MOV DS,AX)**

**; - As DS reserves the number of bytes in the memory space. Hence the value 2000H is now stored in DS)**

1. **Now we can start by moving the values which we want to sort into different locations and the locations should be in ascending order to get the sorted numbers in ascending order . =**

**(MOV [0000H],01H),**

**(MOV [0001H],02H),**

**(MOV [0002H],03H),**

**(MOV [0003H],04H),**

**(MOV [0004H],05H)**

1. **Now we use MOV CH,06H to specify the values till 6.**
2. **Now from here we start our first loop. We are going to use BACK2 as our loop variable here we specify how many times we want to Loop and from Where to start from. By using the commands**

**BACK2: MOV CL,06H**

**MOV SI,000H**

1. **Now after we specified our start location and for how long its time to specify our contents of the loop.**

**We are going to make another variable to Set a loop for our set of instructions our variable will be called BACK1**

1. **Now we are moving the value of SI into AL which was 0000H**

**And setting our starting point.**

1. **Now we are moving the incremented value of SI to AH (A different register) By using the command [SI + 1],AL with this command we will be able to change the location of our pointer to +1 so that we can compare them later.**
2. **Next, we’ll be comparing the values we specified earlier by using the command CMP (it compares any 2 given values)**

**By using the instruction CMP AL,AH**

1. **Now we’ll be setting our instructions to go skip if the returned value of carry is 1. Command = JC SKIP**
2. **Then we’ll be using another command to go to skip if our value is 0.**

**Command = JZ SKIP**

1. **Next, we do the opposite to stop when the compared values are same.**

**By using the COMMANDS MOV [SI+1],AL**

**MOV [SI],AH**

1. **Now here we Start with our SKIP statement from earlier points 10 and 11.**
2. **We start by taking our variable SKIP: all the code inside SKIP will only work when the conditions are met of JC SKIP OR JZ SKIP.**

**Command = SKIP:**

1. **In here we use INC instruction to INCREMENT SI. After which we are decrementing CL which was specified way earlier AT BACK2 .**
2. **And after these commands if the output is not 0 then we say go to LOOP BACK1**

**Command = JNZ BACK1**

1. **Else we are decrementing the Value of CH to get our desired output.**

**And again, we are using JNZ BACK2. To go to loop 2 if the decremented value is not 0.**

1. **Next, we are also going to use the 4CH argument otherwise the command (MOV AH,) will not work we use 4CH to store hexadecimal value.**
2. **Next, we are going to have to use the command INT 21H otherwise our program will not work. As this command is a function dispatcher. It will execute the commands basically.**

**We end our code with CODE ENDS**

**And we also end our START:**

**By END START**

**----------------------------------------Theory Ends----------------------------------------**

**CODE**

**CODE SEGMENT**

**ASSUME CS:CODE**

**START:**

**MOV AX,2000H**

**MOV DS,AX**

**MOV [0000H],01H**

**MOV [0001H],02H**

**MOV [0002H],03H**

**MOV [0003H],04H**

**MOV [0004H],05H**

**MOV CH,06H**

**BACK2: MOV CL,06H**

**MOV SI,0000H**

**BACK1: MOV AL,[SI]**

**MOV AH,[SI+1]**

**CMP AL,AH**

**JC SKIP**

**JZ SKIP**

**MOV [SI+1],AL**

**MOV [SI],AH**

**SKIP: INC SI**

**DEC CL**

**JNZ BACK1**

**DEC CH**

**JNZ BACK2**

**MOV AH,4CH**

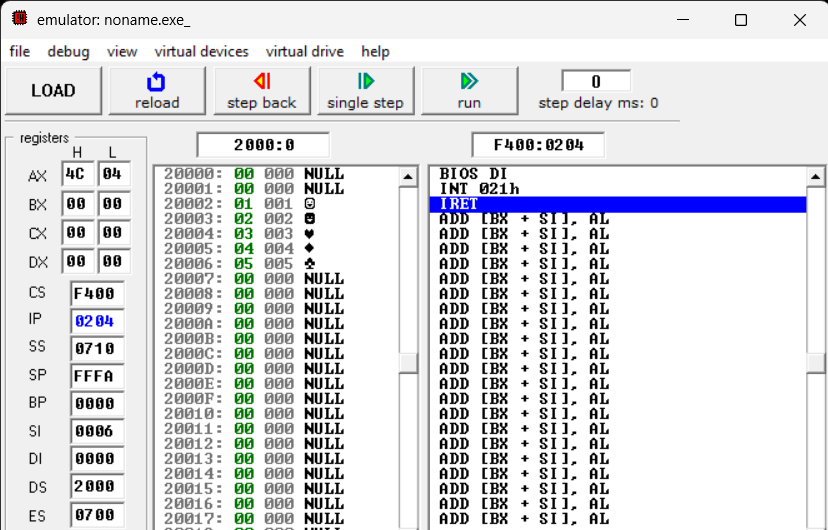
**INT 21H**

**CODE ENDS**

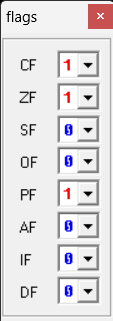
**END START**

**RESULTS/OUTPUT**

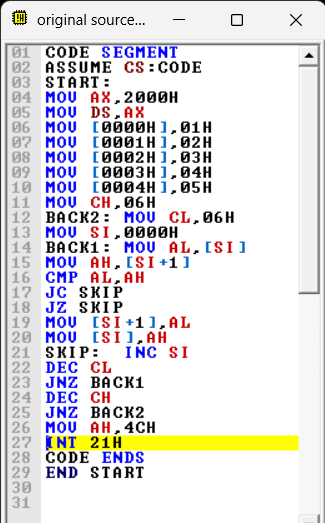
1. **EMULATOR RUNNED**

****

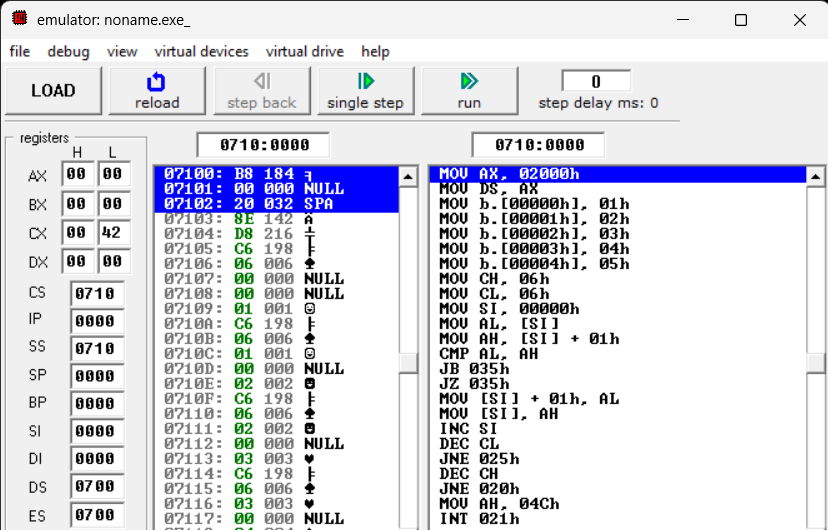
1. **FLAGS**

****

1. **SOURCE**

****

1. **EMULATOR NOT RUNNED**

****